题目要求：
编写程序，能够把如下程序（prog.txt）中的词法单元都识别出来

```
int asd = 0;
int bc = 10;
while ( asd < bc)
{
    if(bc - asd < 2)
        cout<<"they are close."<<endl;
    asd = asd + 1;
}
```

程序说明：

1. 打开 solution.py 文件，确保 prog.txt 和源代码在同一目录下

2. 确保已经安装了 PLY 库

3. 运行 solution.py 文件

4. 对上述 txt 文件中的内容词法分析，解析结果如下图：

```
LexToken(EQUAL,'=',1,8)
LexToken(NUMBER,0,1,10)
LexToken(SEMICOLON,';',1,11)
LexToken(INT,'int',2,13)
LexToken(WORD,'bc',2,17)
LexToken(EQUAL,'=',2,20)
LexToken(NUMBER,10,2,22)
LexToken(SEMICOLON,';',2,24)
LexToken(WHILE,'while',3,26)
LexToken(LPAREN,'(',3,32)
LexToken(WORD,'asd',3,34)
LexToken(LESS,'<',3,38)
LexToken(WORD,'bc',3,40)
LexToken(RPAREN,')',3,42)
LexToken(OBRACE,'{',4,44)
LexToken(IF,'if',5,47)
LexToken(LPAREN,'(',5,49)
LexToken(WORD,'bc',5,50)
LexToken(MINUS,'-',5,53)
LexToken(WORD,'asd',5,55)
LexToken(LESS,'<',5,59)
LexToken(NUMBER,2,5,61)
LexToken(RPAREN,')',5,62)
LexToken(COUT,'cout',6,66)
LexToken(OUTPUT_SYM,'<<',6,70)
LexToken(QUOTE,'"',6,72)
LexToken(STRING,'they are close.',6,73)
LexToken(QUOTE,'"',6,88)
LexToken(OUTPUT_SYM,'<<',6,89)
LexToken(ENDL,'endl',6,91)
LexToken(SEMICOLON,';',6,95)
LexToken(WORD,'asd',7,98)
LexToken(EQUAL,'=',7,102)
LexToken(WORD,'asd',7,104)
LexToken(PLUS,'+',7,108)
LexToken(NUMBER,1,7,110)
LexToken(SEMICOLON,';',7,111)
LexToken(EBRACE,'}',8,113)
```

5. 对程序 token 定义的解释

根据 txt 文件内容，对出现在解析中的 token 做了相关定义。

```
int asd = 0;
int bc = 10;
while ( asd < bc)
{
        if(bc - asd < 2)
                cout<<"they are close."<<endl;
        asd = asd + 1;
}
```

首先是保留字如 int, while 等，使用了一个保留字字典进行存储，由于容易和一般的 word 搞混，添加了类型检查。

```
reserved = {  # 匹配保留字的正确方法，建立一个reserved字典
    'if': 'IF',
    'then': 'THEN',
    'else': 'ELSE',
    'while': 'WHILE',
    'int': 'INT',
    'cout': 'COUT',
    'endl': 'ENDL'
}
```

其次是正常的符号、变量名等。

```
tokens = ["WORD", "STRING", "NUMBER", "QUOTE", "OBRACE", "EBRACE", 'PLUS',
          'MINUS',
          'EQUAL',
          'TIMES',
          'DIVIDE',
          'LPAREN',
          'RPAREN', "LESS", "MORE", "OUTPUT_SYM", "INPUT_SYM", "SEMICOLON"]
```

WORD 代表正常的变量名，STRING 是双引号内的内容，NUMBER 是数字，QUOTE 是双引号”，OBRACE 是左花括号，EBRACE 是右花括号，PLUS/MINUS/EQUAL/TIMES/DIVIDE 分别对应+, -, =, *, /运算，LPAREN 和 RPAREN 分别对应左右圆括号，LESS 和 MORE 对应小于和大于运算，OUTPUT_SYM 对应 cpp 流运算中的<<，INPUT_SYM 对应>>，SEMICOLON 对应分号。

6. 记处理该问题时遇到的关键难点——如何定义匹配的优先级

解决方案：对于正则字符串定义的规则来说，正则字符串越长的优先级越高；对于函数定义的规则来说，哪个放在前面哪个的优先级就高。本题中由于 STRING 中可能包含 WORD，如果 WORD 的优先级高于 STRING，那么 they are close. 这句话就可能被匹配成三个 WORD，造成问题。使 STRING 的优先级高于 WORD 即可解决。其他简单的符号匹配直接使用字符串定义就行，一般哪个长先匹配哪个。

附图：

```
# 通过设置函数的顺序显式地定义优先级！

def t_STRING(t):
    # r'\"[a-zA-Z](.*)\"'
    r'(?<=")([a-zA-Z\s]+[\.])(?=")'
    t.value = str(t.value)
    return t

def t_WORD(t):
    r'([a-zA-Z][a-zA-Z0-9-]*)'
    # t.value = str(t.value)
    t.type = reserved.get(t.value, 'WORD')
    return t
```